Systèmes distribués

TP2: Système distribué à base de courtier de messages

Objectifs

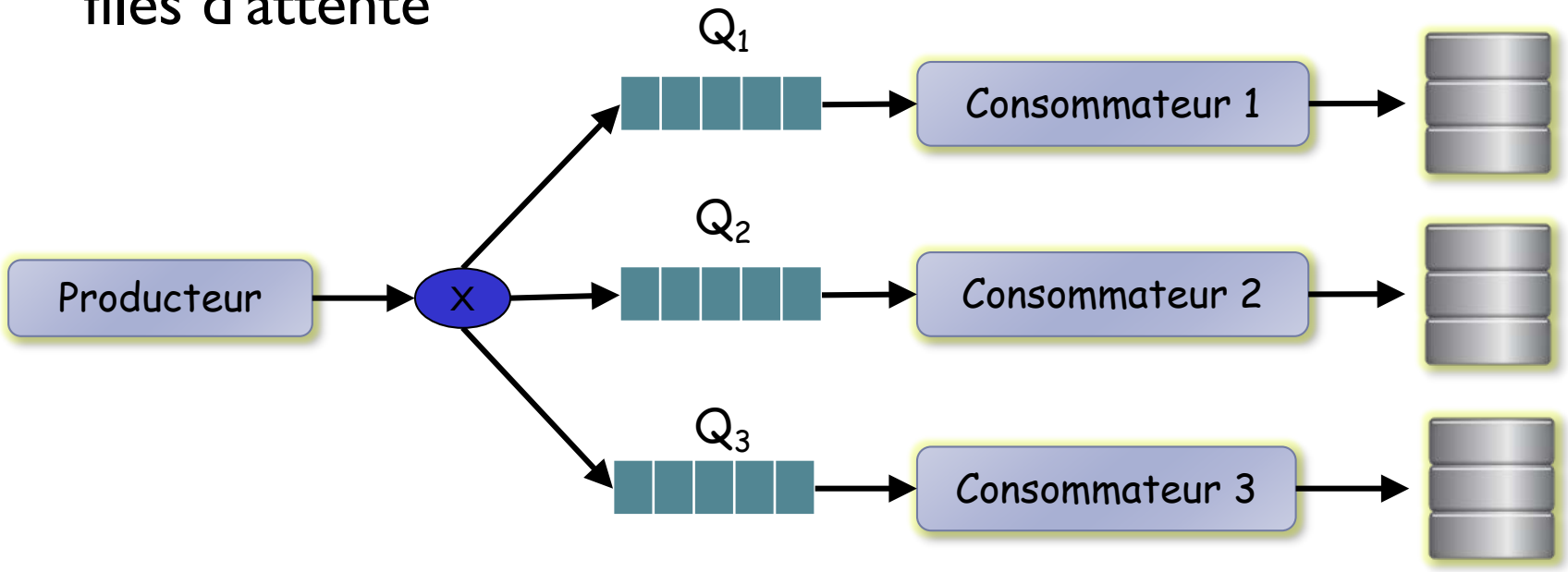
- Développer un système distribué (en cluster) où les tâches sont réparties sur plusieurs nœuds qui communiquent de manière asynchrone en utilisant RabbitMQ

- Avoir un système modulaire pour le traitement de courriels et de traces d'exécution (logs) générées par une application

- Générer des données automatiquement

Travail à réaliser

- Le schéma ci-dessous montre une vue d'ensemble du système à développer
- Ce système est composé de quatre nœuds et d'un modèle de communication avec un échangeur et trois files d'attente



Producteur

- Ce nœud joue le rôle d'une application qui simule la génération de courriels et de traces d'exécution de trois types différents : *avertissement, information, erreur*
  - Les courriels sont routés vers la file d'attente Q_1
  - Les messages de types *information* et *avertissement* sont routés vers la file Q_2 et
  - Finalement les messages de type erreur sont routés vers la troisième file d'attente Q_3

Producteur

- La séparation des traces d'exécution est due au souci de donner une priorité différente aux messages d'erreurs comparativement aux deux autres types de messages

- Les courriels et les traces d'exécution sont générés de manière (pseudo-)aléatoire.
  - Autrement dit, vous allez avoir une boucle (infinie) qui génère les données aléatoirement et la décision, de savoir si c'est un courriel à générer ou un type particulier de trace d'exécution, est prise suivant le nombre aléatoire qui est retourné

Consommateurs

- Le comportement de chaque consommateur est relativement similaire
  - Ainsi, chaque consommateur aura à récupérer un message de la file d'attente à laquelle il est inscrit et
  - Le sauvegarde dans une base de données relationnelle MariaDB
  - Cette base est composée de deux tables *Courriels* et *Logs* ayant un schéma qui correspond au format des messages échangés (voir le point qui suit)

Format de données

- Les données sont échangées sous format JSON comme suit :
  - Courriels : un message JSON ayant les attributs suivants *From*, *To*, *Subject*, *Body*
  - Trace d'exécution : un message JSON ayant les attributs suivants *Type*, *Date*, *Body*

Cluster

- La base de données doit être mise en cluster multi-master en utilisant par exemple MariaDB Galera Cluster
- RabbitMQ doit être déployé en mode cluster avec des files d'attente en miroir.

Environnement recommandé

- Systèmes d'exploitation:
  - Linux (Centos 7.1)
- Langage de programmation:
  - Java ou tout autre langage qui est supporté par RabbitMQ (Ruby, Python, etc.)
- Courtier de messages:
  - RabbitMQ
- Base de données:
  - MariaDB

Évaluation

- Les éléments suivants seront pris en considération lors de l'évaluation de ce travail:
  - Étapes du TP:
    1. Fonctionnement sur une seule machine
    2. Fonctionnement en mode distribué (idéalement un programme par VM, mais peut tolérer deux programmes par VM)
  - Un rapport expliquant:
    - La démarche
    - Le schéma de base de données relationnelle proposée
    - Les problèmes rencontrés et les solutions proposées
    - Le rôle de chaque classe/programme
    - Comment déployer et lancer les différents modules de votre système
    - Tableau récapitulant la contribution (en pourcentage) de chaque coéquipier sur les différentes parties du TP
  - Une démonstration du système
    - Une séance sera réservée aux démonstrations
  - La qualité du code
  - La complétude du système
  - L'implication des coéquipiers (gestion de code source avec GIT et GitLab est obligatoire et les *commits* doivent être réguliers et impliquant les différents coéquipiers)
  - La capacité à retrouver l'information pertinente et nécessaire à la réalisation du travail pratique

Barème

- Producteur : 2 points
- Consommateur 1 : 1,5 points
- Consommateur 2 : 1,0 point
- Consommateur 3 : 1,0 point
- Mise en cluster: 1,0 point
- Qualité du rapport : 1,0 point
- Qualité du code et de la présentation : 2,5 points

Consignes

- Travail à effectuer en équipes de 3 personnes
- À remettre sur Moodle par un seul membre de chaque équipe:
  - Fichier zip/7z/rar contenant le code de votre système et le rapport
- La version sur GitLab doit être aussi à jour
- Dernier délai pour la remise du travail:
  - Le 17 avril à 10h00